School of Electronic Engineering and Computer Science

# Final Report

**Programme of study:**

Computer Science with Business Management

## Project Title:

Reservation Database System with Analysis

**Supervisor:**

Dr Pengwei Hao

**Student Name**:

Chieh-Yu Chou

Final Year
Undergraduate Project 2016/17

Queen Mary
University of London

Date: 24/04/2017

# Abstract

This project was designed to improve the current existing restaurant booking system as well as analysis the booking records. In other words, the end system will not only directly interact with the diners who wants to book the table but also the manager/administer can use it as a useful tool to analysis booking records. The end system will be divided into three categories, Book a table, Edit booking and Analysis (Login required). First, two categories are designed for the restaurant consumers, the basic functionality of booking system will be included, in addition, there will be few extra functions added to the system such as additional requests and the waiting list. An analysis is designed for the restaurant managers which allows them to breakdown booking records in monthly, seasonal, yearly and represent the data in graphical format.

In order to achieve the project's aims, review current systems was fulfilled not only get a general idea of design the booking interface but also what functions must include in the end system. The questionnaires were designed to identify what features must and least important to contain in booking system, even more, it indicated new features they would like to included in the booking system. Similarly, analysis functionality was carried out by review existing system and noted down important features as well as possible features to add into end system. With all the feedback received, requirements was next step before going into the design process. It identified the potential functional and non-functional requirements that I should included to my system while designing and coding.

Next part of the report illustrated how the end system interface was designed that followed by analysis existing system and list of functional and non-functional requirements suggested from the questionnaire. Moreover, the report discussed the details of the problems and solutions had come up while developing end system.

After end system fully coded, testing performed to make sure everything is running smoothly and one more questionnaire was taken by few users who used this end system. This will not only provide the direct usage experience feedback of system but additionally, the suggestions on layouts, representations and functions helped me to improve it.

# Table of contents

# Figures

# Tables

# Chapter 1: Introduction

## 1.1 Project Aims

The purpose of this project is to improve the interaction between system and consumers and implement unique features. In this case, the research of the current existing systems are important, also a well-design questionnaire is required in order to capture what consumer thinks about booking system and with lists of potential improvements. Consequently, the project required a well-structure database to store customer information, booking records and staff details. Additionally, this project will not only have the basic restaurant reservation booking system but with the analysis tool built-in with the system. The analysis will only be used by the manager, it will require to login to the system as manager then fetch the booking records from the database and convert those text or numeric data into graphical results and some calculations will perform such as most popular month this year. This will provide not only directly but immediately breaking down numeric statistics into graphical formate.

List of aims:

- Improvement on the current existing restaurant reservation booking system.
- Well-structure and design for both questionnaire and database.
- The end system must be easy and simple to use for all user(booker/manager/staff).
- Adding the new and improve features into the system.
    - Online/up-to-date waiting(queuing) list.
    - Estimate waiting time (Online waiting list).
    - Send confirmation Email with confirmed booking details after the user made online reservation.
    - Analysis system for the manager, with different categories.
        - Diners, number of diners entered this year or month.
        - Months, number of records(booked) this year or month.
        - Times, number of diners entered in different time.
    - Graphical results in different formate.
        - Bar chart.
        - Line graph.
        - Pie chart.

        - Area chart.
    - Ranking system (data are comparable).
        - Compare results in different years.
        - Display those results on same graph.
    - Searchable (data are searchable).
        - Staffs and managers are able to access booking records and search specific record easily.

## 1.2 Motivation and challenges

According to the statistics, there are 3,696,238,430 internet users till 2016 and currently the internet usage is still growing (Internet World Stats, 2017) [1]. While the internet usage is growing, simultaneously the online booking reservation system usages are increasing. Many restaurants have their own website and a reservation system for their diners to book a table in just few steps, if the tables are fully booked then the diners will find the next available restaurant. However, if there is a system can either optimise bookings or tell consumers the next available table time, which will not only keep consumers but more importantly it brings positive effect on the profits. I named it "Online self-queue" system, system will estimating current the dining time for each table depended on number of diners and calculating the waiting time for each required table depended on number of diners. For example, a table required for two diners will have short waiting time then a table for ten diners. Additionally, system only available when the current section (lunch or supper) are booked out, diners can join the queue before entered restaurants.

The major challenges for this project will be having a well-structured database, the accurate waiting time and data analysis. In order to complete those challenges, self-learning advanced techniques in SQL and Java are the other challenges. Furthermore, a decent planning of time and project management are very important because during the developing of final project, there are other coursework deadlines need to work on.

## 1.3 Overview

This report has eight sections, the introduction is section one which stated the aims, motivation and challenges for this project. Section two delivers the research conducted before developing project with background explanations for the readers who may has or has not basic knowledge about reservation system, database system, structured query language, data analysis and JavaFX. Next section defines the functional and non-functional requirements for this project and use case examples. Forth and fifth section shows the design and implementation for this project, design section included diagrams when is necessary, implementation section tells the problems and potential solutions has came up while developing project. Last two sections discuss what I have achieve and further development of this project.

# Chapter 2: Research

## 2.1 Background

**Restaurant Reservation**

There are three potential ways that restaurant consumers can reserver a table, phone reservation, online reservation or visit restaurant to make the reservation. Online reservation can be done at any time and anywhere consumers where as phone and in-door reservation only available during restaurant opening hours. Even though, there are pros and cons for each reservation process, but they have one common, record the booking details. Either the system or the staff when they reached consumer's reservation query, the process of record booking details are very important. A well-structured record system will affect tracking and analysing record much easier.

**Database System**

The end system will require a Database Management Systems (DBMS), which not only enable the restaurant consumer to book the table but also the restaurant manager can review the data. DBMS is a collection of programs that enables you to store, modify, and extract information from a database (Vangie Beal, 2005) [2]. A well-tailored database or table structure can increase data analysis performance, which allows the manager team to break down data more easily. The DBMS provides users to create, retrieve, update and manage data (Margaret Rouse, 2015) [3]. Once the database is been built, the end user will be the restaurant manager team. By allowing end user to edit the records in the database are very important, for instance, end user can update booking details if the consumers entered were wrong.

**SQL**

At early 1970s, Dr. Edgar F. "Ted" Codd introduced Relational Database Management System model, around 1974 SQL appeared. Structured Query Language (SQL) is a computer language for retrieving, storing and updating data stored in a relational database. It is the common language for Relational Database System.

SQL can be divided into three sections:

- **Data Manipulation Language (DML)** - This can be used as store, update and remove data with the INSERT, UPDATE and DELETE statements.

- **Data Definition Language (DDL)** - This is used to organising tables and index structures by the following statements: CREATE, ALTER, TRUNCATE and DROP.
- **Data Control Language (DCL)** - This is commonly used to set the user's permissions by using the GRANT and REVOKE statements.

In order to maximise the functionality of my project, I have used the 2-Data Languages mentioned above (DML and DCL). The end users (bookers and managers) are able to insert, update and delete the booking records, for the managers they can change the staff permissions, such as allowing staffs to add new staff account.

## Data Analysis

In terms of data analysis, it is the process of extracting, modelling and paraphrasing the numeric data in order to generate the key information for the company, which means a proven way for organisations and enterprises to gain the information they need to make better decisions (Molly Galetto, 2017) [4]. For example, the restaurant managers can access those information as supporting materials to set the goals for next the months or year. There are two types of data analytics methodologies, exploratory data analysis (EDA) and confirmatory data analysis (CDA) (Margaret Rouse, 2016) [5]. EDA is used to find the similarity and differences between each data set, whereas CDA is determined true false and tested a hypothesis of a data set.

## JavaFx

JavaFx not only allowed the developers to create the applications across multiple platforms but also provided the richer user interface design, i.e CSS can be applied to the control UI. As the diagram Figure 1. demonstrates the JavaFx architecture divided into nine sections. Prism is used to do the rendering job so FX has better graphic performance than other language. such as Swing. JavaFX APIs provided rich packages for developer to program with, for instance, Javafx.scene.chart supports different chart components for data visualisation.

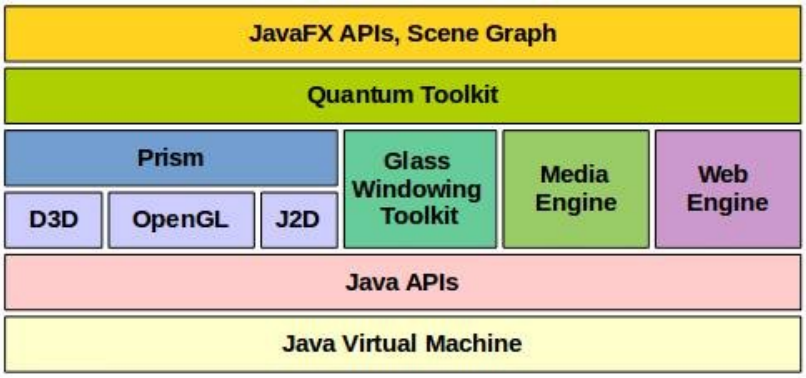



**Figure 1.** JavaFx architecture (Manoj Debnath, 2016) [6]

**JFoenix**

JFoenix is one of the open source Java library that supports JavaFX application, as well as it has greater design of java components. As the Figure 2. illustrates JFoenix has more stylish buttons design then JavaFX, by adding the JFoenix library to my application can definitely brings the interface design up to the next level.


**Figure 2.** JavaFx (left) and JFoenix (right) buttons styles

**CSS**

Cascading Style Sheets, known as CSS, they are the technical specifications for a layout (Jennifer Kyrnin, 2016) [7]. Normally, the web developers will included the CSS to there web page for the better graphic layout, however, JavaFX has allowed CSS to be applied to the application.

## 2.2 Review existing system

Moving into the research and review current restaurant reservation systems, I had compare and contrast those systems with my project. The Table-1 below has selected five restaurant's reservation system compared with my project, there is a list of ten functions that are a basic reservation system should be included. One of the restaurant I have selected, Yauatcha they do not have their own reservation system. They used third-party reservation system called "OpenTable" to help them manage consumers reservations, rest of the restaurant selected had a direct reservation system.

Next, Table-2 illustrates the comparison between my project and third-party reservation system, I picked the top four famous online table reservation websites that people usually used to book or find a restaurant.

**Table-1 Review Existing Systems**

| Restaurant name | Duck & Waffle | Kintan | Steak & Co | Yauatcha | The Breakfast Club | My Project |
|---|---|---|---|---|---|---|
| **1. Edit booking** | - | YES | - | - | - | YES |
| **2. Number of booking steps** | 4 | 5 | 2 | 3 | 3 | 3 |
| **3. Maximum diners allowed for booking** | 8 | 14 | - | 20 | 15 | 10 |
| **4. Notice maximum table time** | YES | YES | - | YES | - | YES |
| **5. Limited months accepted for booking** | YES | YES | YES | YES | YES | YES |
| **6. Add additional request** | YES | - | YES | YES | YES | YES |
| **7. Online self-queue system** | - | - | - | - | - | YES |
| **8. Receive confirmation via EMAIL** | YES | YES | YES | YES | YES | YES |
| **9. Estimated next available table** | - | - | - | - | - | YES |
| **10. Rich user interface design** | - | - | - | - | - | YES |

**Table-2 Review Third-Party Reservation System**

| Name | OpenTable | Bookatable | Quandoo | ResDiary | My Project |
|---|---|---|---|---|---|
| **1. Edit booking** | - | - | - | - | YES |
| **2. Maximum diners allowed for booking** | 20 | 6 | 8 | 8 | 10 |
| **3. Notice maximum table time** | YES | - | - | - | YES |
| **4. Limited months accepted for booking** | YES | YES | - | - | YES |
| **5. Add additional request** | YES | YES | - | YES | YES |
| **6. Online self-queue system** | - | - | - | - | YES |
| **7. Receive confirmation via EMAIL** | YES | YES | YES | YES | YES |
| **8. Estimated next available table** | - | - | - | - | YES |
| **9. Rich user interface design** | - | - | - | - | YES |

## 2.3 Research conclusion

As the research from Table-1 conducted, five current existing systems only one system has included the edit booking function, other system will require bookers to contact the restaurant directly if they want to update their booking details.

## 2.4 Summary

This chapter has conducted the research conclusion, review current existing systems, including third-party restaurant reservation systems. I had also explained what is JavaFX and why is a well-structured database is important.

# Chapter 3: Requirements specification

## 3.1 Primary functional requirements

The requirements identified at this section are the key components I must included to the system.

- Table reservation interface: The rich graphic user interface design is required and provide the useful tips when are necessary. In addition, the end user should be able to access to other pages easily, i.e switching to editing booked or online self-queue interface. Users should get the listed of instant feedbacks:
    - If users picked date, time and number of diners, the system should immediately check the table availability for that time. Pop-up an fully booked dialog if no available slot, otherwise proceed to the next step.
    - If users missed to fill any fields, an message and highlight that particular field is required.
    - The system should allowing user to preview their booking details before they submitted.
    - The system should including a progress bar with instantly updated bar so user can know how many steps left.
- Editing booked reservation interface: User only needs to enter phone number and booking confirmation code to view, update or cancel their booking.
- Online self-queue interface: When user entered this section, the system should automatically check the current date and dinning period, if the tables are fully booked, then user will able to access this section. Meanwhile, if it is not booked out, it should alert user with the message and redirect to the booking section. When the user entered this section, they should be able to notice how long will they need to wait for the next available table and how many diners are in the queue. Finally, system should allow users to cancel self-queue request, once they request to cancel they will be removed and queue list should automatically refreshed.
- Admin login interface: This part should only contains two input fields and two buttons. Additionally, user should be able to change to others scenes easily.
- View database interface: The managers or staffs are able to view today's bookings and all bookings (past and upcoming) with the consumers bookings details.

- Display graphical interface: Chart should be display largely with clearly labelling x-axis and y-axis.

## 3.2 Secondary functional requirements

Once I had complete all the primary functional requirements, I will implemented secondary functional requirements.

- Animations: The transitions between scenes are used to interact with the users.
- Validation the input fields:
    - First name and surname: The system should check the input values are non-numeric context.
    - Contact number: The system should only allowing numeric values to be entered.
    - E-mail address: The system should validate the input values are in correct e-mail formate or not.
    - Date and time: Only present and future dates and times can be selected.
- Send confirmation email: After users confirm their booking details and submitted, an confirmation email should be sent to them immediately. Email contents should including:
    - Diner's first name and surname.
    - Booked date and time.
    - Number of diners booked for a table.
    - Contact number.
    - Confirmation code (Booking reference).
    - Additional requests.
    - A greeting message.
- Logged in admin (user) authentication: The system should authenticate the logged in user is manager or staff. Depending on logged in as staff or manager, the view database interface should be differently.
    - If logged in as staff:
        - Staff should be able to view today's bookings.
        - Staff should be able to view all bookings.
        - Staff should be able to add or update consumer's booking details.
    - If logged in as manager:
        - Manager should have same functions that staff had.

        - Manager can add or update staff details.
        - Manager can update table settings, i.e. change maximum table time.
        - Manager can access to graph report section.
- Database records are searchable: Both booking records and staff records should be able to search.
- Specific chart report: The manager should be able to enter the year or month the want to see on the chart.
- Combine multiple timeline in one graph: The manager should be able to compare the results in yearly, for example, 2015 to 2017 illustrated on same graph.

## 3.3 Database requirements

- The select statement, this statement should be applied when:
    - The system wants to check some data in the database whether are existing or not.
    - The system wants to output the data into text fields.
    - The system wants to display the data into the table view.
    - The system wants to output the numeric data into a graphical formate.
- The insert statement, this statement should be applied when:
    - The systems received the user's input value and want to store into the database.
- The delete statement, this statement should be applied when:
    - The user is requesting to delete or cancel booking records.

## 3.4 Non-functional requirements

- Efficiency:
    - Booking steps: As the research conducted, the average booking steps are 3.4 steps, so the system should allowing the user to complete the booking task within 3 steps. Entering data, time, number of diners and personal information are the first step, then system should displays the details entered and allowing user to add additional request, finally system should shows the booking reference (confirmation code) and send the confirmation email to user when user submitted book details.
    - Edit booked steps: The system should enable user easy and quick access to manage their booking details, maximum of 3 steps to complete the task. First step, enter

booking reference and contact number, then switched scene to display the booking details if found. User can choses to update their booking information or cancel the booking, finally system will output a confirmation message.

- Online self-queue steps: When the user entered to this scene, they should be able to join the queue, cancel the queue and see the waiting list and approximate waiting time at same page.

- Stability: The system should run smoothly without any failing while the user is interacting with system. In order to achieved this, I should test system carefully and noted down the errors appeared during testing.
- User Interface:
  - Colours and backgrounds: Some of the reservation systems I reviewed are hard to read the text and message, therefore careful selection of colours and backgrounds are required.
  - Text font: The size of text font should be selected carefully so the user can capture the content easily.

## 3.5 Use case diagrams

The use case diagram summarise all the functionalities of the system with the user friendly representation. The system includes three use cases, booker, waiting diner and admin. As Figure 3. shown, the booker can search an available table or edit booked table with options update booking details or cancel booking. Those options, reserve a table, update and cancel booking will eventually modify the database.

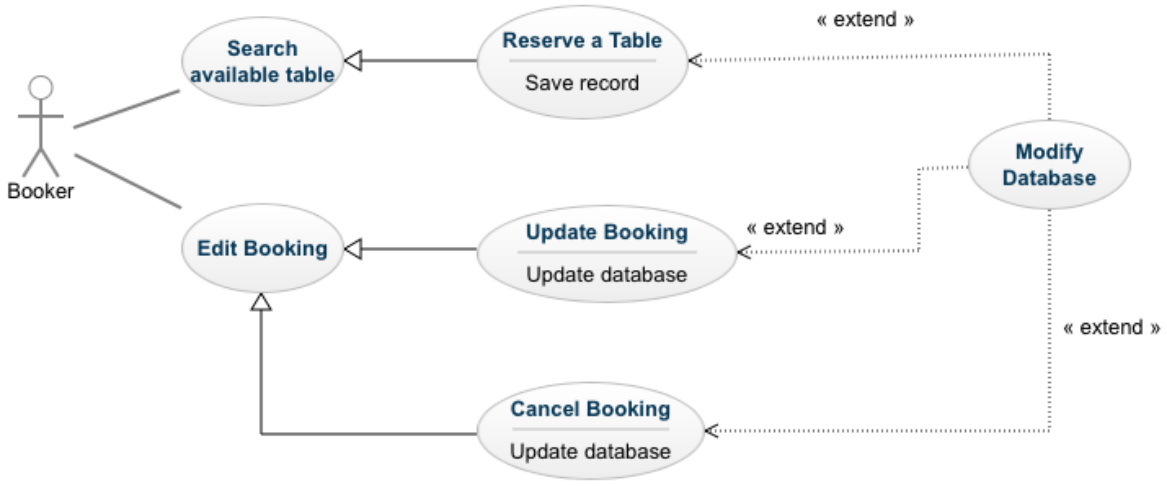



**Figure 3.** Use case diagram for Booker

The waiting diner case diagram (Figure 4.) has defined two cases, join queue or cancel queue. When the user chosen one of these case the queue list will be updated.

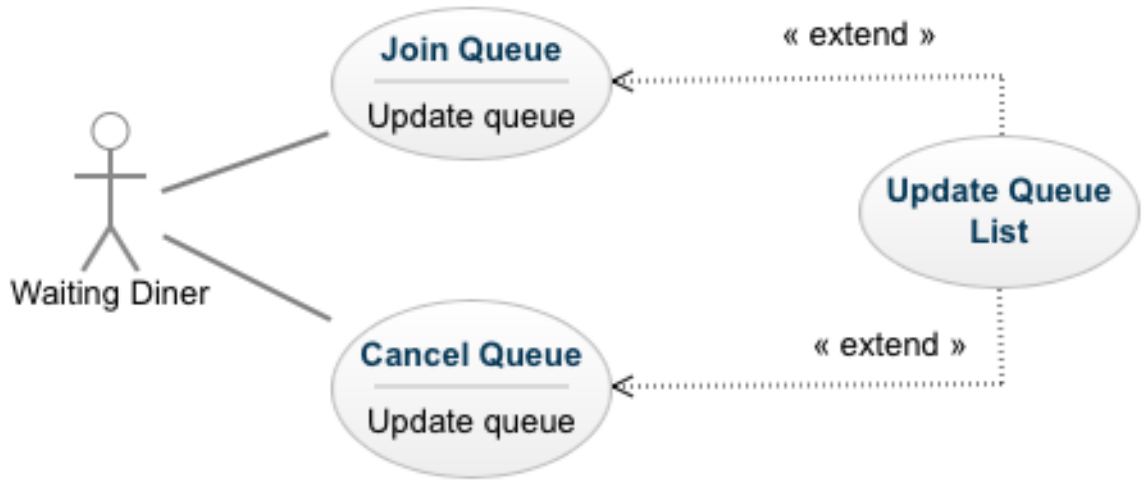



**Figure 4.** Use case diagram for waiting diner

Third use case diagram (Figure 5.) illustrates five potential actions that admin can perform, however before they can access to those actions, a login with verification is required. If the user logged in has been verified as staff, user only can view booking records, modify records and search records. The other two actions, view staff records and data analysis will be available when user logged in as manager.

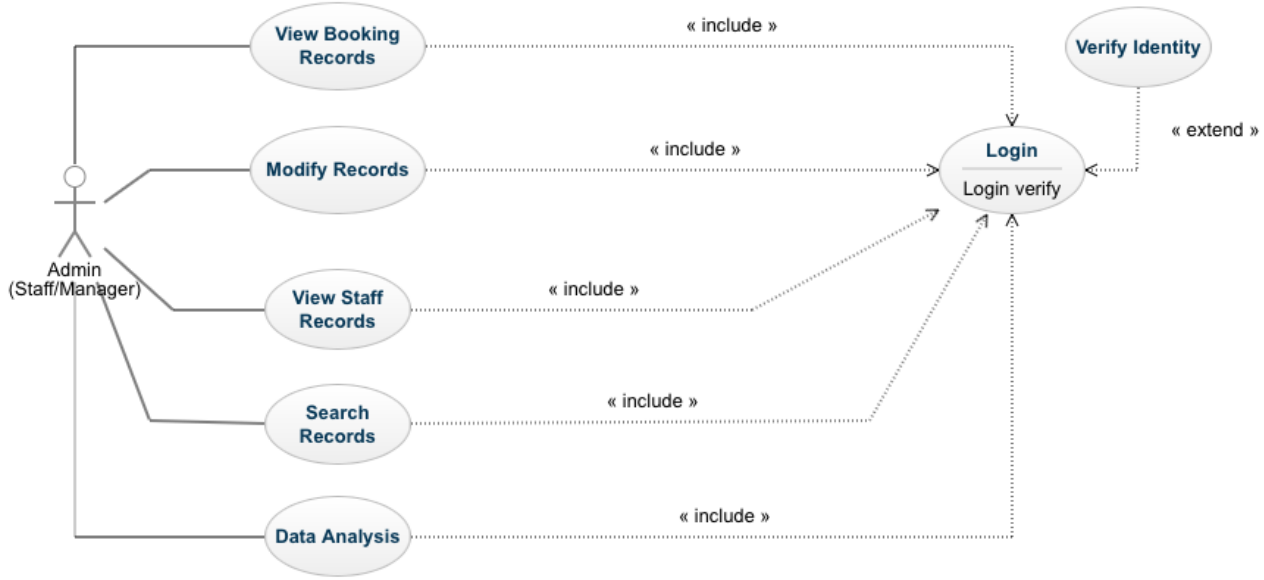



**Figure 5.** Use case diagram for admin

## 3.6 Summary

This chapter defined the functional, non-functional and database requirements for my system. Primary functional requirements are the functions that I must implemented before I moved on to secondary functional requirements. Database requirements are very important as well, as I mentioned in Chapter 2. Background that a well-built database with using correct query to execute the data can save time consume. Once I have completed those three requirements, I will be working on non-functional requirements. Beside those requirements, during implementation process I must carefully follow through the use cases diagrams.

# Chapter 4: Design and Implementation

## 4.1 Development environment

I used Gluon SceneBuilder to built the JavaFX FXML and NetBeans with the JavaFX 2.0 Plugin to implement the system. Gluon SceneBuilder is free and open source with integrated the JavaFX ecosystem official controls. In addition, I added the JFoenix Library to SceneBuilder, which enabled me to drag and drop the stylish and beautiful designed components into the system.

## 4.2 High level design

The MVC design pattern included Model, View and Controller (Figure 6.), it is not only used to differentiate the system concerns but also it provide a good system structure. Model represents as data or message carrier that will update the controller if the data has been changed. View represents the data carried by model and user can send input to the controller. Controller is the link between model and view, the data will flow into model and update the view.

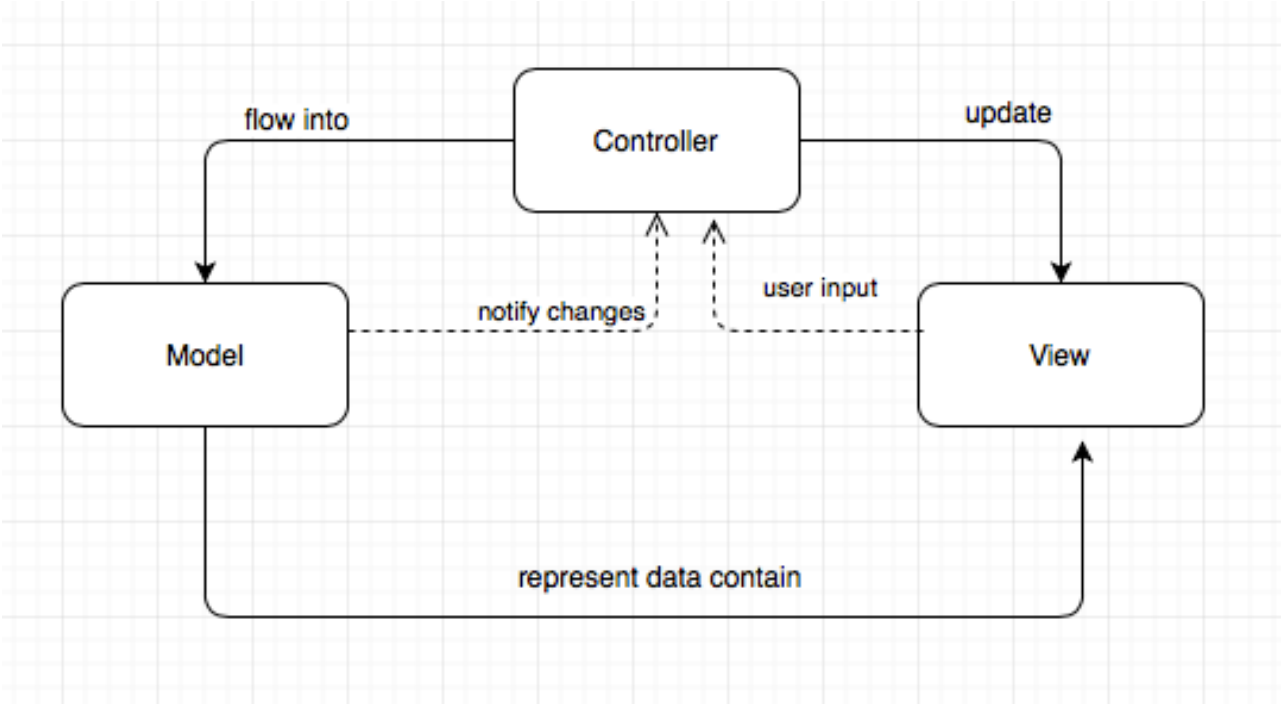



**Figure 6.** MVC design pattern

### 4.2.1 Model package

The model package contains classes, which are used to fetch the data then deliver to the view-controller package. For instance, customer's first name or staff's contact number. The view-controller package will change the data if the user entered new, edit or remove data. For example: a user who booked a table at 19.00 p.m but want to change the table time to earlier so the value entered and controller received the user input and passed to model and update the view.

### 4.2.2 View-Controller package

This section consists view and controller package, top level Main_Interface is view package linked with six controller package. he view-controller package is shown below (Figure 7.). Main_Interface uses to display each interface with suitable data for that section when the user required. Each controller will gathering the user input values and send it to the model package to store the values and update the view package with latest information.

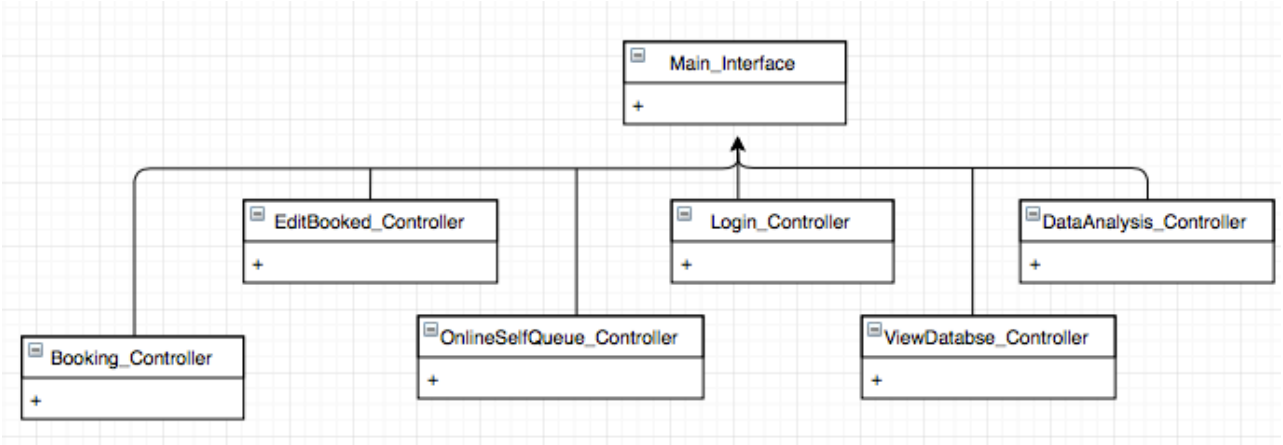



**Figure 7.** View-controller package high-level class diagram

## 4.3 Low level design

This section will breaking down each package with steps by steps how functions worked and discussed the decision I while implementing the system.

### 4.3.1 Model package

This model package consists classes which used for store data and at a certain stage the data will be modified by controller package and display the data via view package.

- customerList: This class represents the customer booking record with first name, surname, date, time, number of diners, contact number, e-mail, additional request, period, and booking reference. Depending on user's request method, customer booking details can be output as a list view, table view or text fields. For example: The user entered view database section assume user has logged in, the model package will passing the customer booking records to view-controller package and records will be displayed in a table view. I chosen table to display the data at this section is because it is easier for the admin to view data with clear interface.
- staffList: This class represents the staff personal information with first name, surname, contact number, contact address, date of birth, role, username, and password. Similarly, staff information can be shown in different form, like the customerList class.

### 4.3.2 View-controller package

This package consists of user interface elements, user interaction components and the components to display the information sent by model package.

- Main_Interface: This package has provided the transitions between each scene. For instance, if there user is currently at book a table scene and switched to edit booked scene, during the switching an transit animation will be played.
- Booking controller: This package is designed for the user to book a table with a date-picker time-picker, a combo-box stored number of diners ("1,2,3,4,5"), textfields for user entering personal details and a progress bar has been implemented at the footer. After user submitted the booking query, the system will get the newest e-mail address inserted to the database and send an confirmation e-mail with confirmed booking details to to that user.
- Edit booking controller: This package allows the user who wants to change or cancel their booking by entering the contact number and booking reference. Then this package will send these two information to the model package to find any record is matched, if matched record found, model package will sent the booking details (first name, surname, date, time, number of diners, additional request) back to this package and displayed each details. There

are two button update booking and cancel booking. No matter user click either one of them, model package will receive the message “update” or “delete” that booking record.

- Online self-queue controller: This package will estimate the approx table waiting time and display today's restaurant consumers who is currently in the queue. As an illustration, a user uses the system, first the view package (Main_Interface) applied and user change interface to online self-queue, at this stage, the Main_Interface will find suitable data and display it. If the user enter their phone and number of diners, online self-queue controller package will send the details to model and stored into database then update the queue list.
- Login controller: There are two text fields, username and password for admin to insert the validate value. Below the password text field has two button, login and cancel. If the user hit the cancel button, the Main_Interface will change with transition. The account authentication will take place when user hit the login button.
- View database controller: After logged in, the login controller passed the user id and role to this package, package will then use it to determine whether the user is manager or staff. There is a table to display all the booking records. User can edit the data by selecting the row and system will display that data in text fields. The button “Data analysis” will be visible only when the logged in user is manager.
- Data analysis controller: This package contains large graph (chart) at the top and sets of button at the bottom. Each button has it’s own ID which the system can use those ID indicating the type or graph (Bar chart, Line graph, Pie chart, Area chart), type of data want to be display onto the graph.

## 4.4 Screen flow

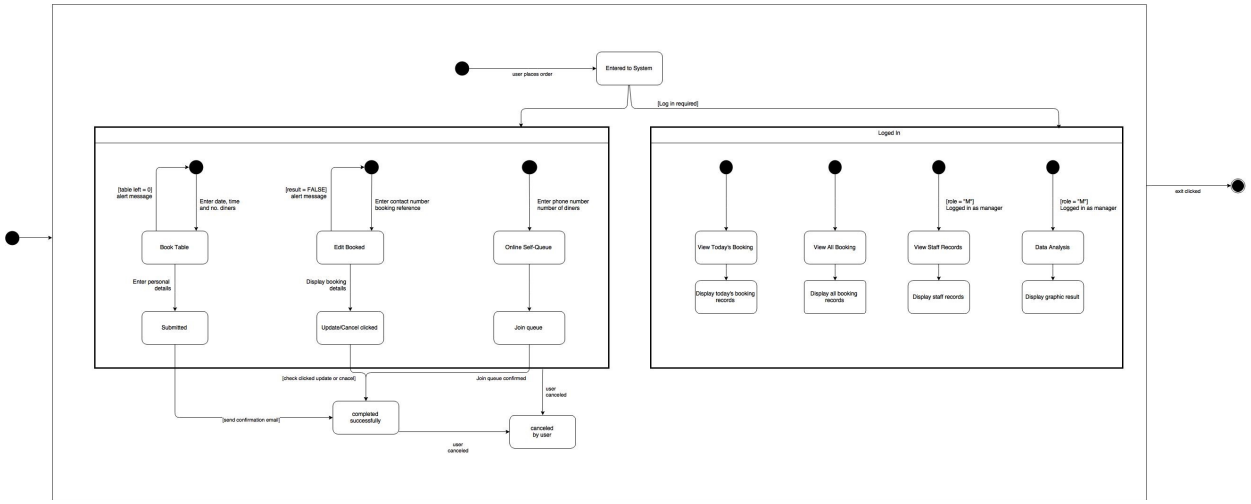



**Figure 8.** System screen flow

The system has seven main screens with eight sub-screens, the user launched system will automatically direct to the “Book a table” scene (Figure 8.). Each main screens and sub-screens are separated by a state. On the left hand side is the main screens for the restaurant consumer’s, the right hand side is required log in to access those screens.

## 4.5 Application walk through

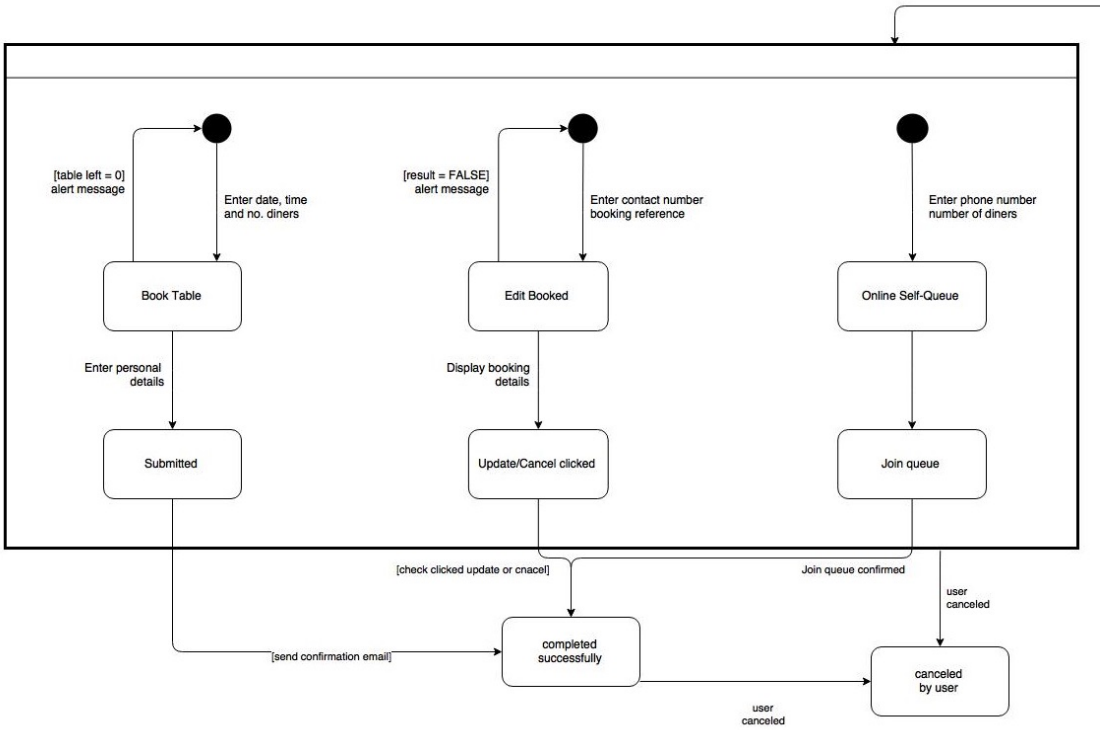



**Figure 8-1.** System screen flow (left part)

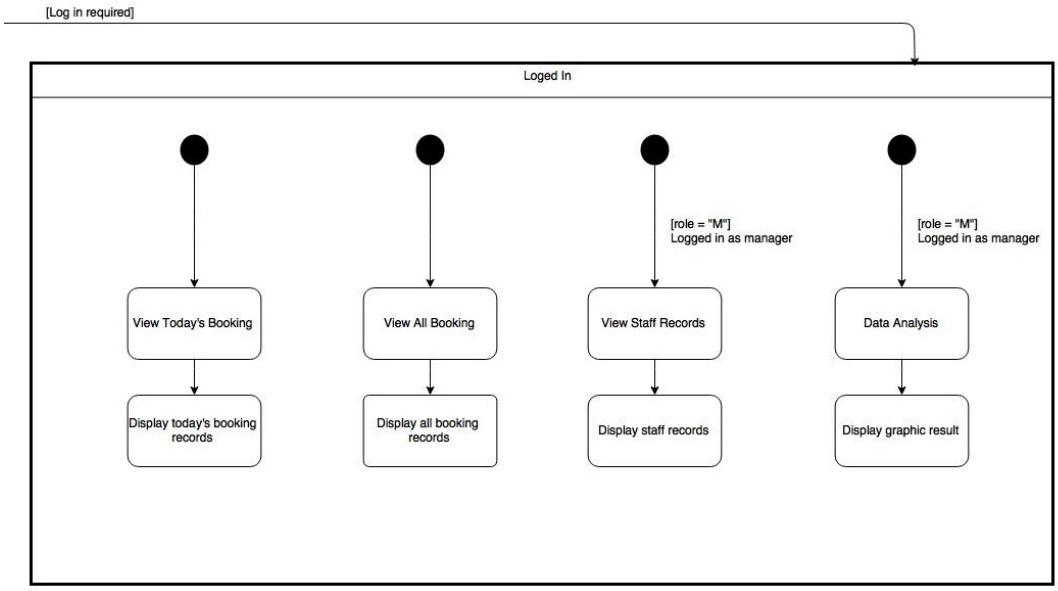



**Figure 8-2.** System screen flow (right part)

The application walk through for each screen is provided with a screen flow and screen stages, followed with this order, screen flow: top left, stage 1: top right, stage 2: bottom left, stage 3: bottom right.

**Booking screen**

This is the first screen, Booking scene (Figure 9-1.), system requires to select date first then number of diners then time, at this stage system will check the availability for that date, time and number of diners requested. If it is available, user will need to enter their details in order to click "Next" button. The progress bar at the bottom updated depend on the number of fields entered correctly, as shown progress bar has been filled up between each stage. Once user entered details correctly, next stage printed out the details they entered and additional requests can be added or leave as blank. When user confirmed click the "Confirm" button proceed to next scene which displayed the confirmation code and booked date, simultaneously the code has sent to user via the e-mail address they provided.

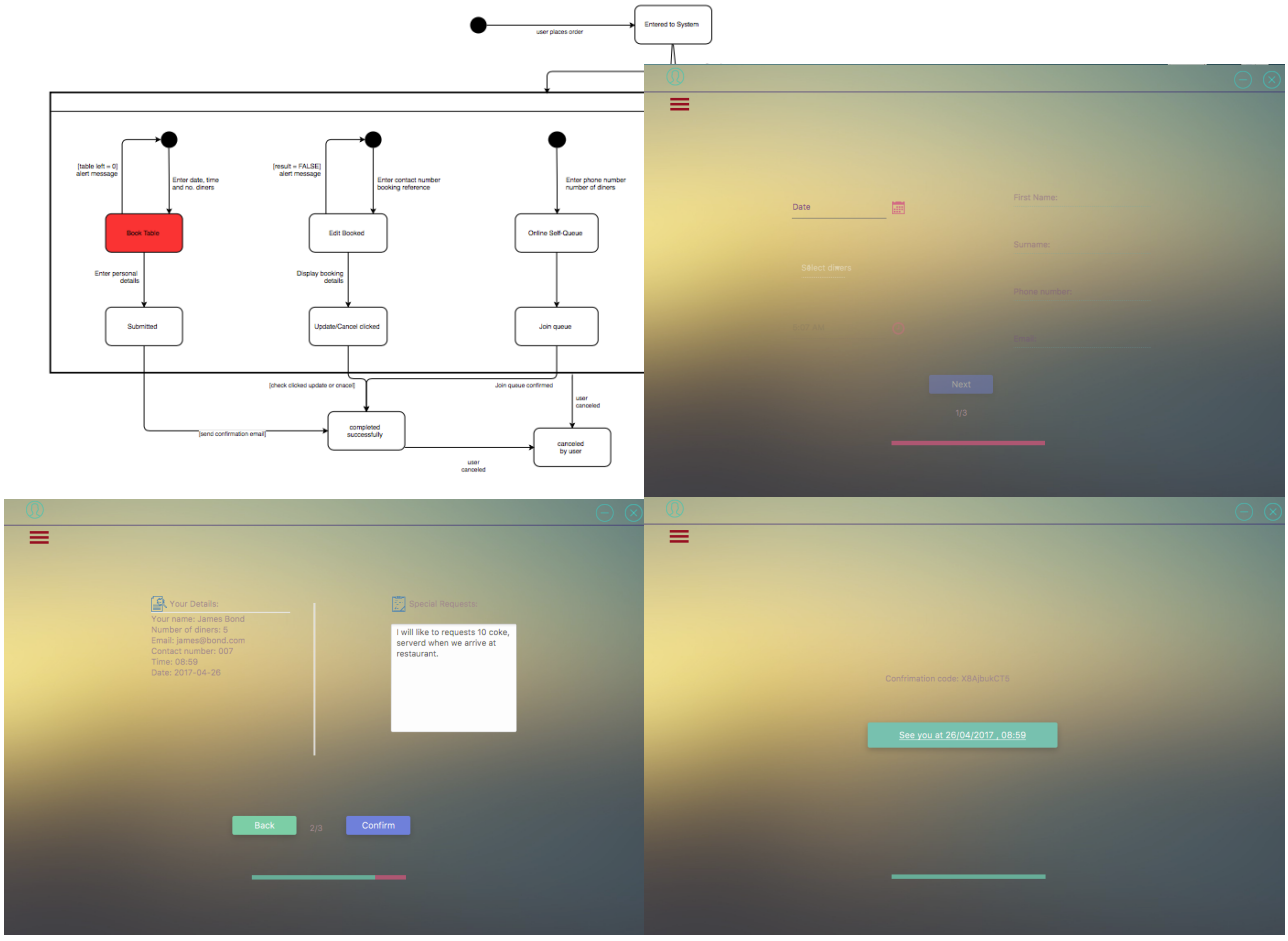



**Figure 9-1.** Booking screen

**Edit booked screen**

This is the edit booked screen (Figure 9-2.), at first the system requires the user to enter their contact number and booking reference (confirmation code) to change their booking. If the phone number and code matched to the database record, system will redirect to stage 2. However, if one of the value did not found, system will alert users. Stage 2, user will see their booking details with first name, surname, e-mail address, contact number, number of diners booked, date and time. At this stage, user can change either update the booking ("Update Booking" button default set as disable since there is no change detected) or cancel their booking. No matter which buttons they clicked Stage 3 will shown with the text "Booking information has been updated.".

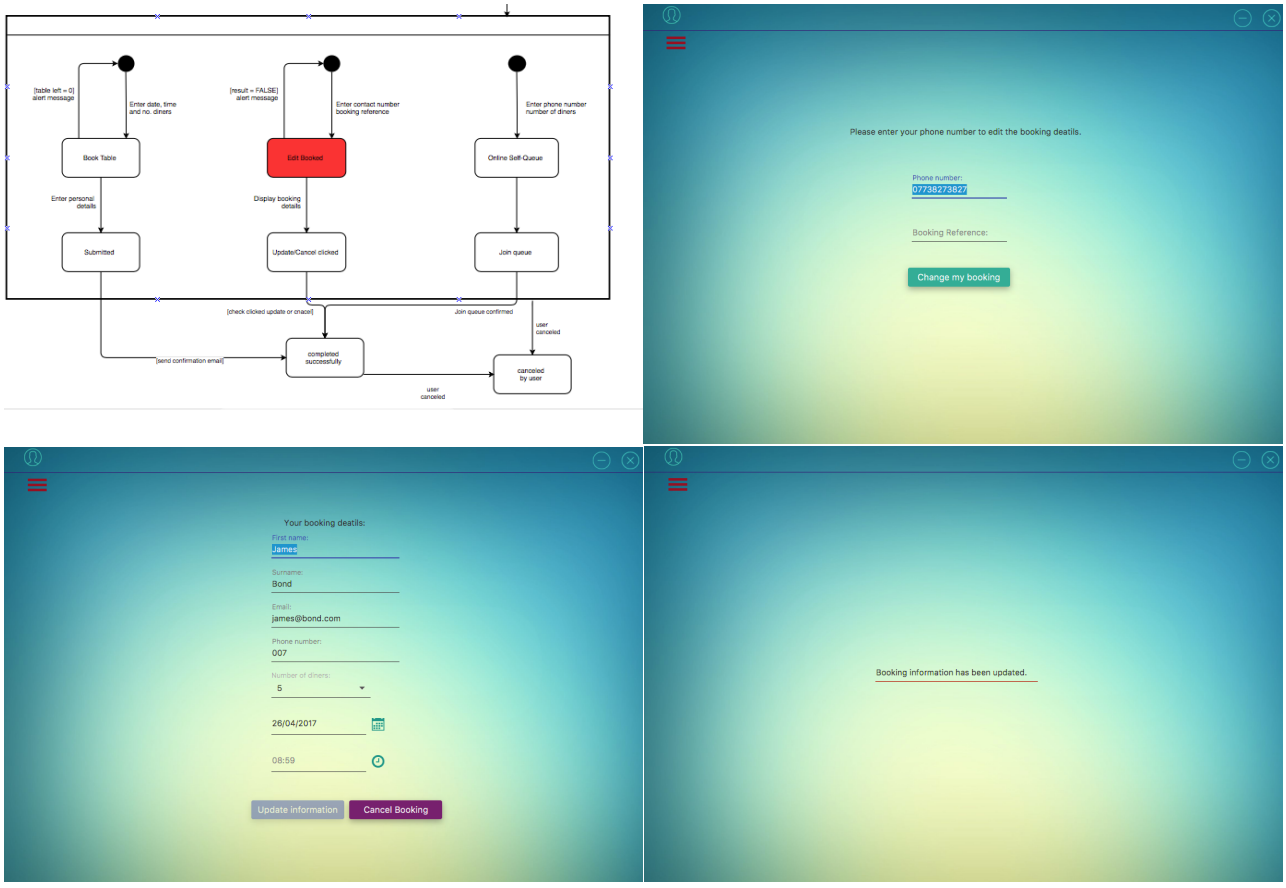



**Figure 9-2.** Edit Booked screen

## Online self-queue screen

This is the online self-queue screen (Figure 9-3.), it had two list-views, four input components and two buttons. First list-view, estimating waiting time list at top left of the online self-queue screen is used to display approximate table waiting time and number of people in the queue in five different lists, table for 1 - 2 diners, 3 - 4 diners, 5 - 6 diners, 7 - 8 diners and 9 - 10 diners. Second list-view is queue list, used to show the queue number followed by name and number of diners request. User can enter their name, contact number and number of diners to join the queue. Or, user can cancel the queue by entering contact number that they registered to join queue. The queue list and estimating waiting time will automatically updated when there is a changed applied to system.

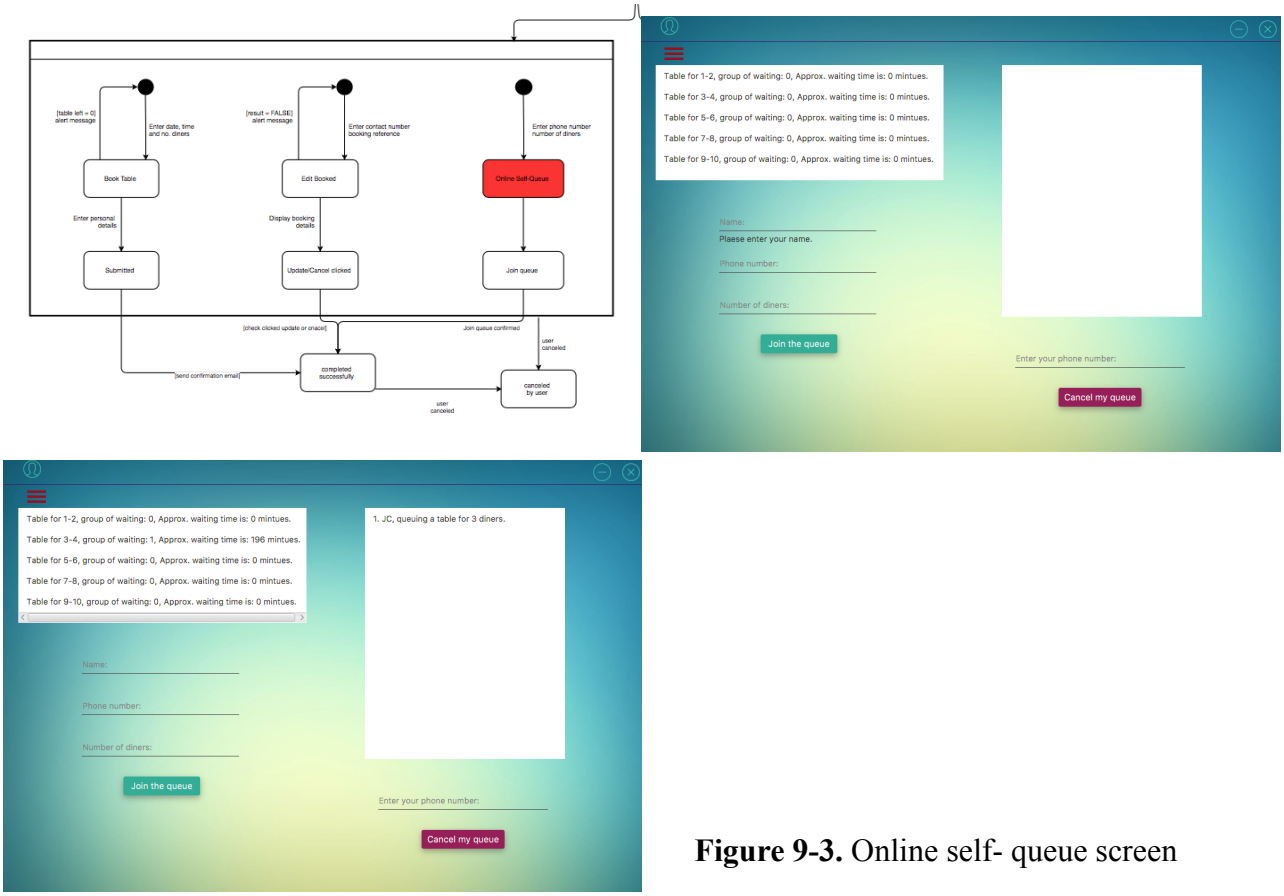



**Figure 9-3.** Online self- queue screen

## Header bar and side menu

The system had included three different header bars and one side menu (Figure 9-4.). One header bar is initialised when the system lunched, it consists of three buttons, close, minimise and admin login. Second header bar is loaded when user clicked admin login button, it contains close, minimise and home (re-direct to booking screen) buttons. Third header bar is if admin logged in, this header contains close, minimise and view database buttons and a welcome back text message included the username. The menu bar initialised as close, when user open it, it has three buttons, book a table, edit booked and online self-queue. The menu bar is initialised though the entire system.

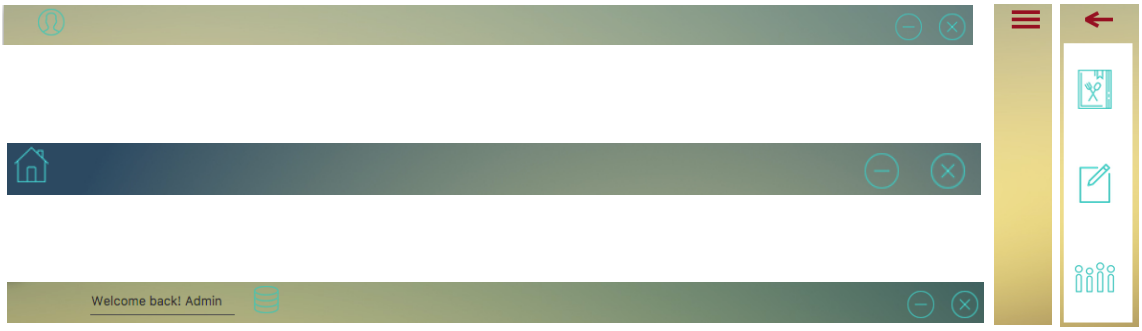



**Figure 9-4.** Header bar and side menu

## Login screen

This is the login screen (Figure 9-5.), it consists of two input components and two buttons. If user has successfully logged in, system will automatically re-direct the screen to view database screen. However, if user has failed attempt to login, system will display a login error message.

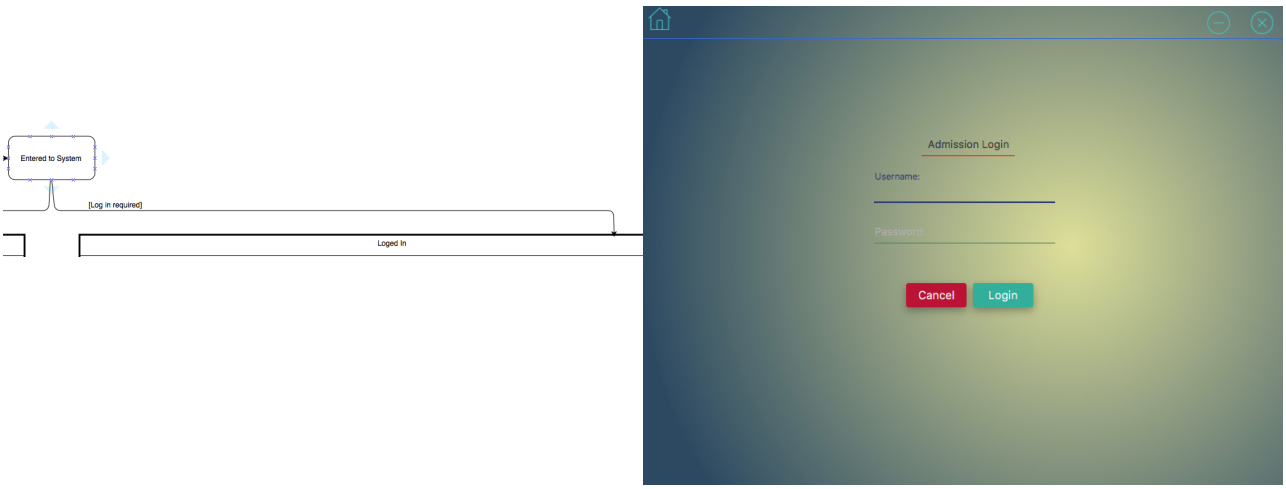



**Figure 9-5.** Login screen

## View today's booking records screen

This is the view today's booking records screen (Figure 9-6.), it has one input component, one table view and five buttons (three buttons if logged in as staff), the admin can easily change the table view to view staff records (if logged in as manager) or view all booking records. If admin had selected one the row in the table and clicked "Update Record" button, system will reached to database and find that booking records then displayed on to screen. Admin can either choses "cancel" button to go back or "Update record" button to modify that booking record.

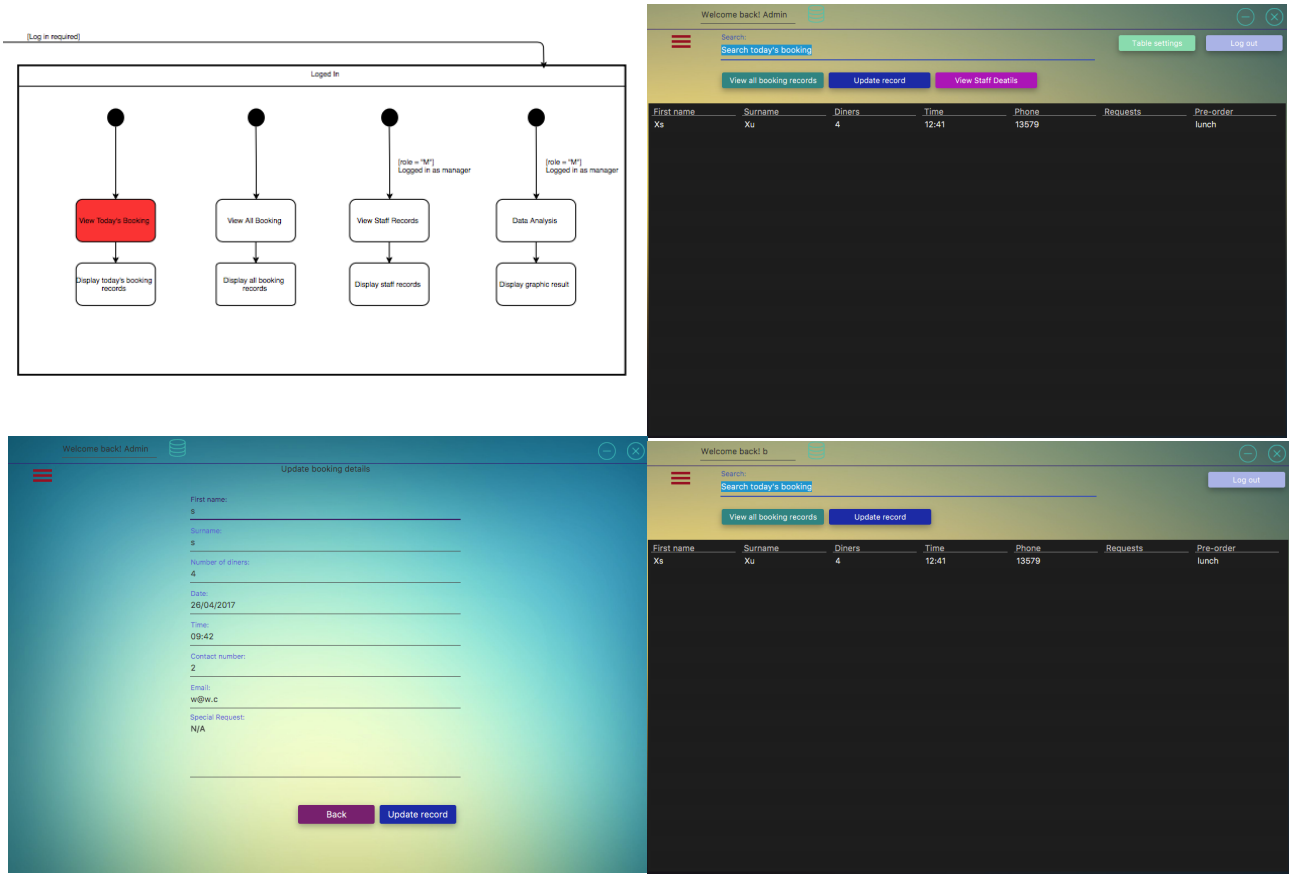



**Figure 9-6.** View today's booking records screen

## View all booking records screen

This is the view all booking records screen (Figure 9-7.), it has one input component, one table view and six buttons (three buttons if logged in as staff), the admin can easily change the table view to view staff records (if logged in as manager) or view today's booking records. If admin had selected one the row in the table and clicked "Update Record" button, system will reached to database and find that booking records then displayed on to screen. Admin can either choses "cancel" button to go back or "Update record" button to modify that booking record. The input component functioned as search data, admin can type in text and system will find the matching data with that text and update table view. Once admin cleared the text input, table view will display all the contents again.

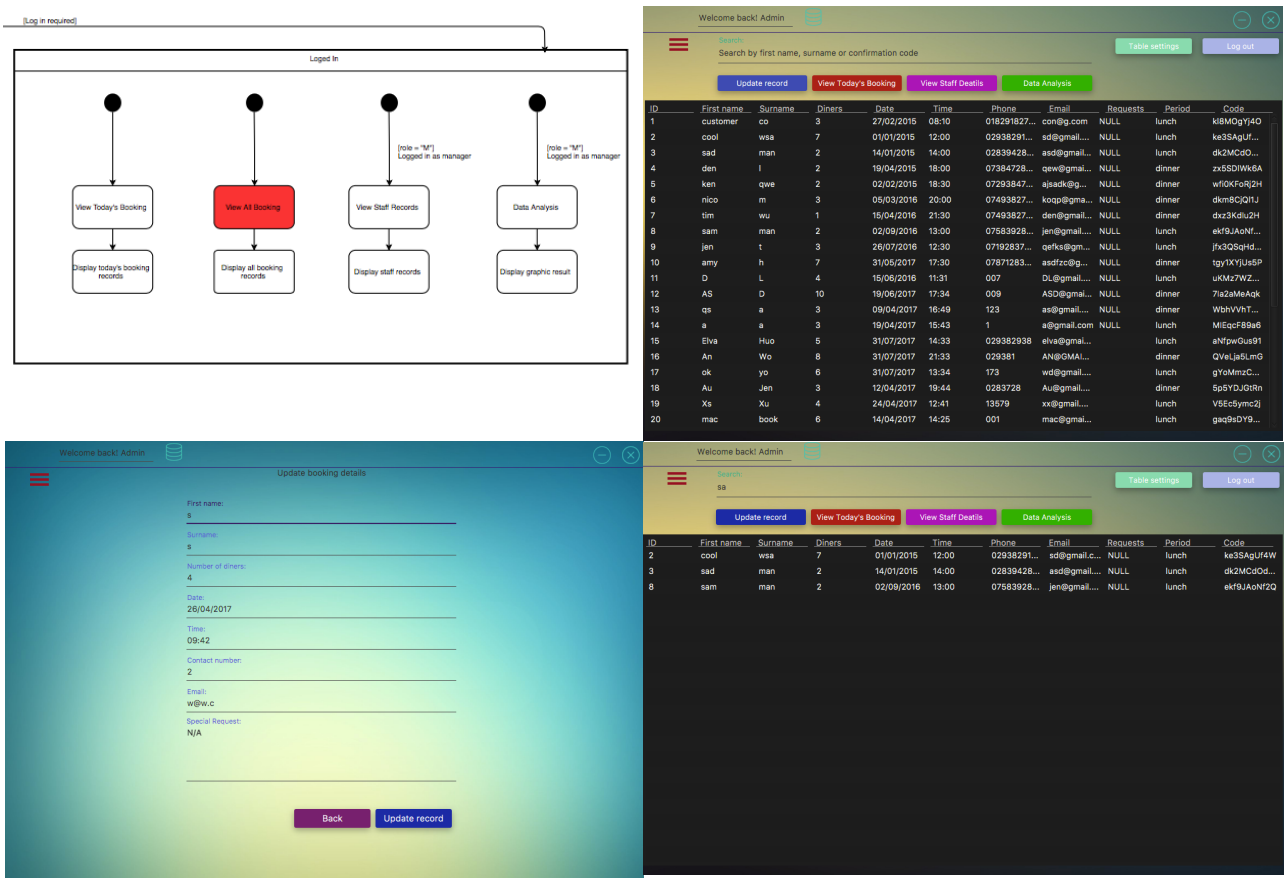



**Figure 9-7.** View all booking records screen

## View staff records screen

This is the view staff records screen (Figure 9-8.), it has one input component, one table view and seven buttons (six buttons if logged in as staff), the admin can easily change the table view to view all booking records or view today's booking records. If admin had selected one the row in the table and clicked "Update Record" button, system will reached to database and find that staff records then displayed on to screen. Admin can either choses "cancel" button to go back or "Update record" button to modify that staff record. The input component functioned as search data, admin can type in text and system will find the matching data with that text and update table view. Once admin cleared the text input, table view will display all the contents again. Additionally, admin can select one row and click "Remove an account" to remove that account (staff details).

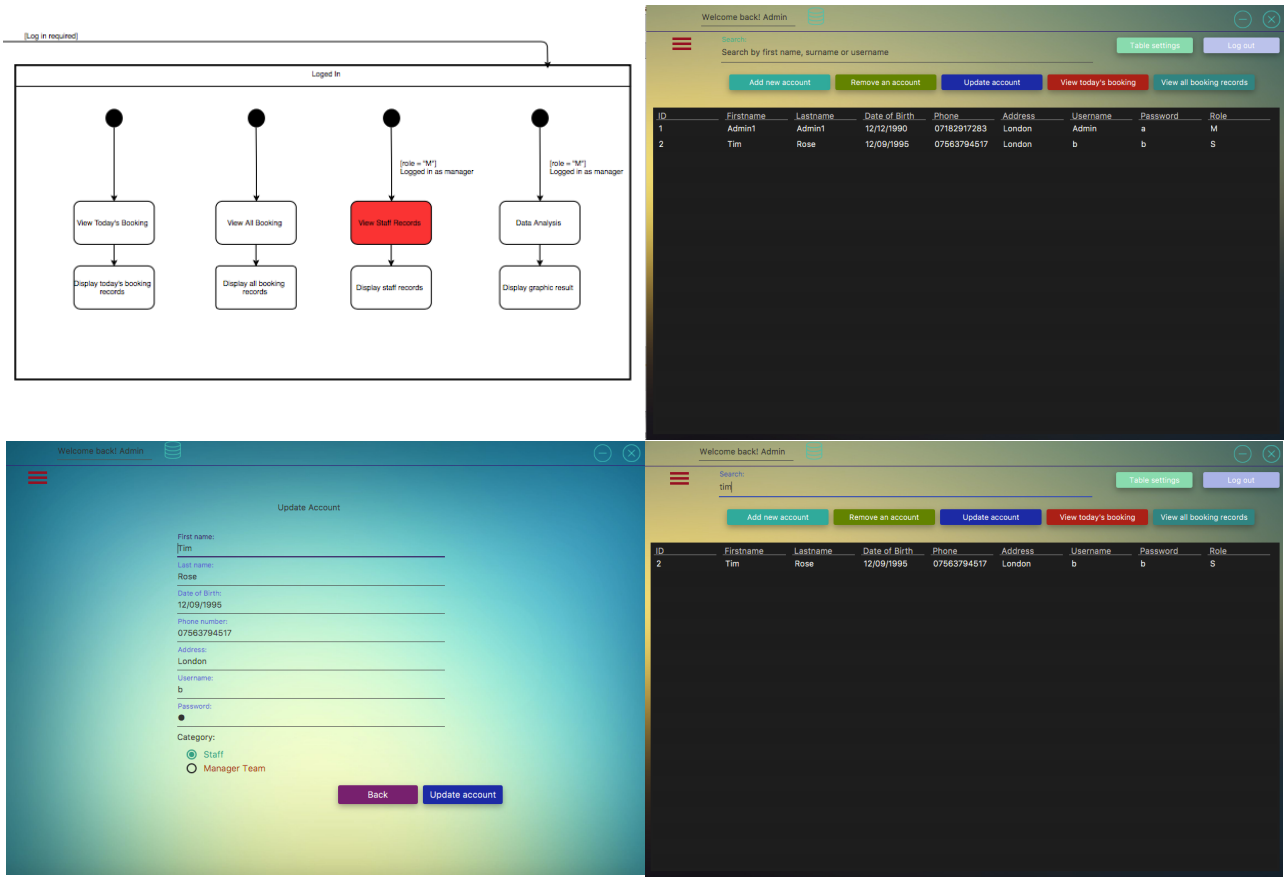



**Figure 9-8.** View staff records screen

**Data analysis screen**

This is the data analysis screen (Figure 9-9.), only admin logged in as manager can access to this screen. This screen contains three set of buttons at bottom left of screen, first set of buttons included “Diners”, “Month” and “Time (Period)”. First set is used to categorise the data type, “Diners” means number of diners booked by monthly, “Months” means number of records booked by monthly and “Time (Period)” means number of diners accessed to the restaurant at what time. Second set of buttons included different types of graph, line graph, bar chart, pie chart and area chart. Last set of buttons included an range year (2014 to 2017), current year, previous year and the year before previous year button. Those button have enable manager to choose different type of result they want to see in different type of graphs with the year.

Four input components under those button sets are used for manager to search particular year, months and an range year (2014 to 2017). Once the manger typed in then click the “Generate” button to view the graph, and of course  manger can choose different type of graphs and categories.

On the left hand side, it calculated a quick statistics that maximum, minimum and average number of diners booked this year. There is a “Back” button at the bottom right screen to switch back to view database screen. The middle of the screen is where the graph will be shown.

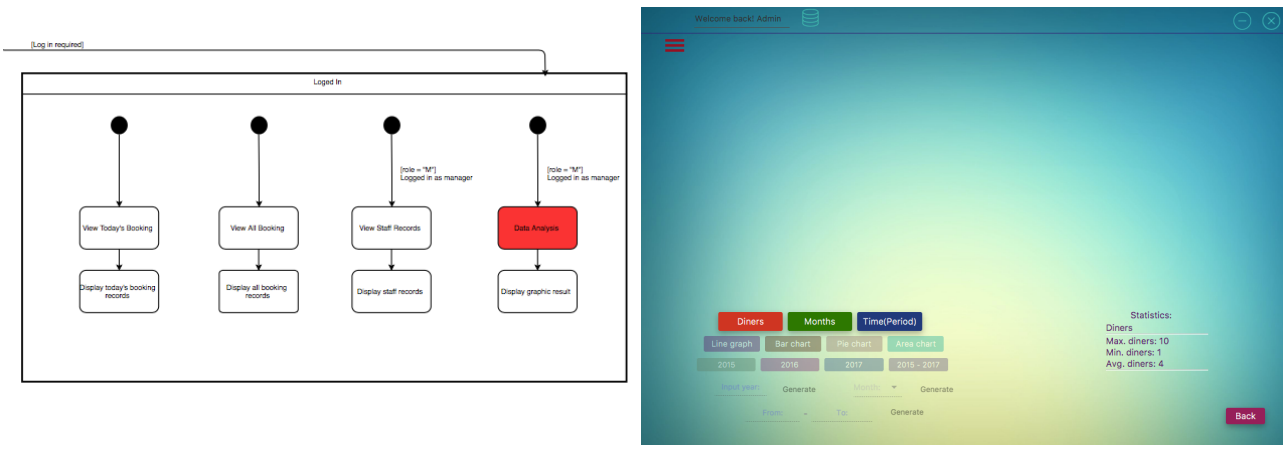



**Figure 9-9.** Data analysis screen

## Line Graph

The line graphs (Figure 9-9-1.), first line graph shown that total number of diners booked by monthly through out the current year. Second graph shown the total number of diners booked from 2012 to 2016 as manger  had inputted the value and generate the graph. I believe this type of graph will help manager with data comparison and it is easy to spot which month is the peak month as well as when did the record drop compare to previous years.


**Figure 9-9-1.** Line graphs

## Bar Chart

The bar charts (Figure 9-9-2.), first bar chart shown that total number of booking records booked by monthly through out the current year. Second graph shown the total number of booking records from 2015 to 2017 as manger by clicked the button “2015-2017”. I believe this type of chart will help manager with data comparison and it is easy to spot which months are the peak month as well as when did the record dropped compare to previous years.


**Figure 9-9-2.** Bar charts

## Pie Chart

The bar charts (Figure 9-9-3.), first bar chart shown that total number of diners booked by monthly through out the 2015. Manger can enter the year manually as well, just like previous graphs. This type of chart had a clean image as the the system will divide the records. I believe manager can use pie chart to manage quick access of breaking down diner booked or number of booking records in few seconds.

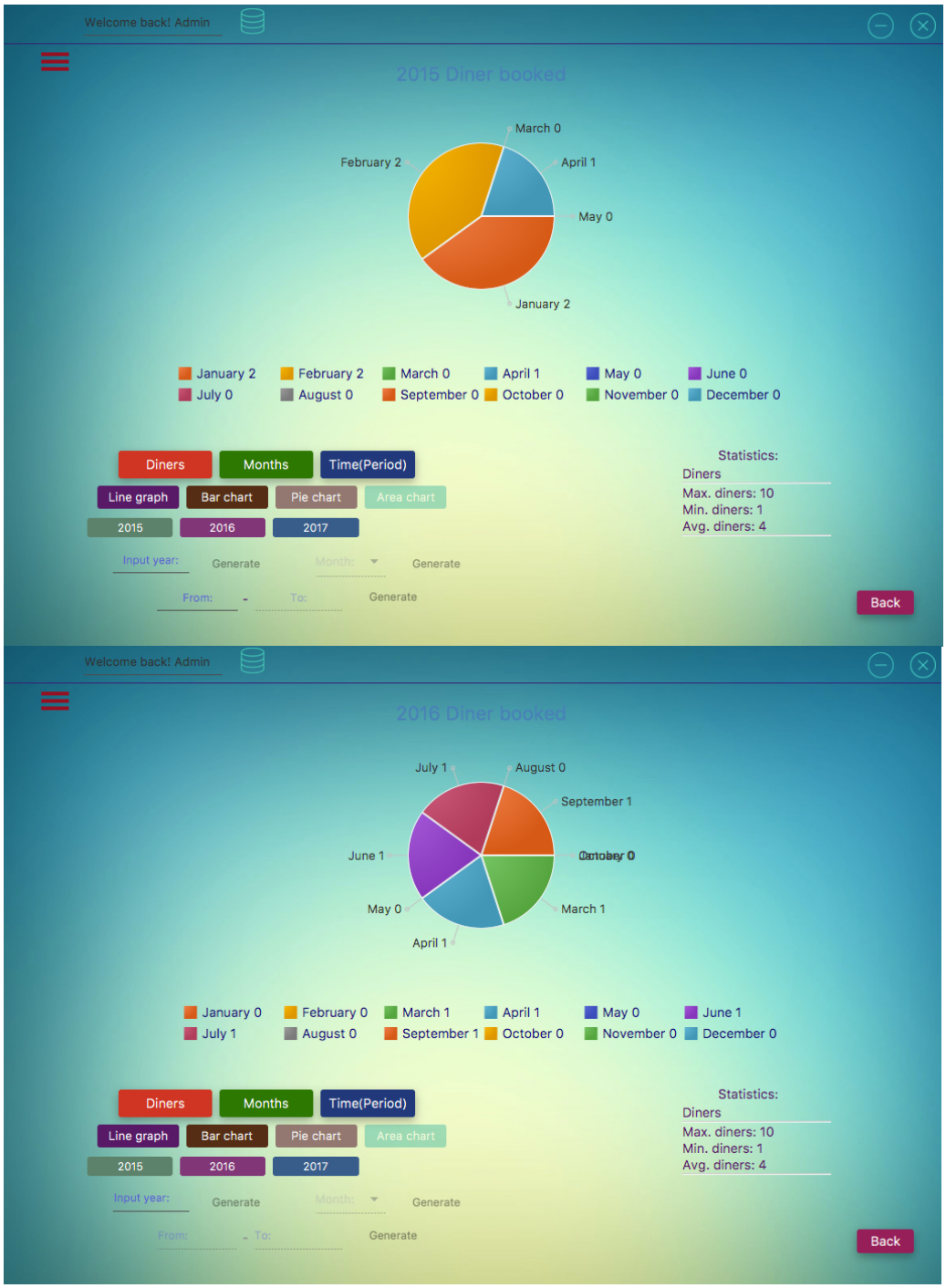



**Figure 9-9-3.** Pie charts

## Area Chart

The bar charts (Figure 9-9-4.), first area chart shown that total number of diners booked by period (24 hours) through out the current year. Second graph shown the total number of diners booked from 2015 to 2017 as manger by clicked the button “2015-2017”. I believe this type of chart will help manager with data comparison and it is easy to spot which period if the day are the peak time as well as when did the record dropped compare to previous years.

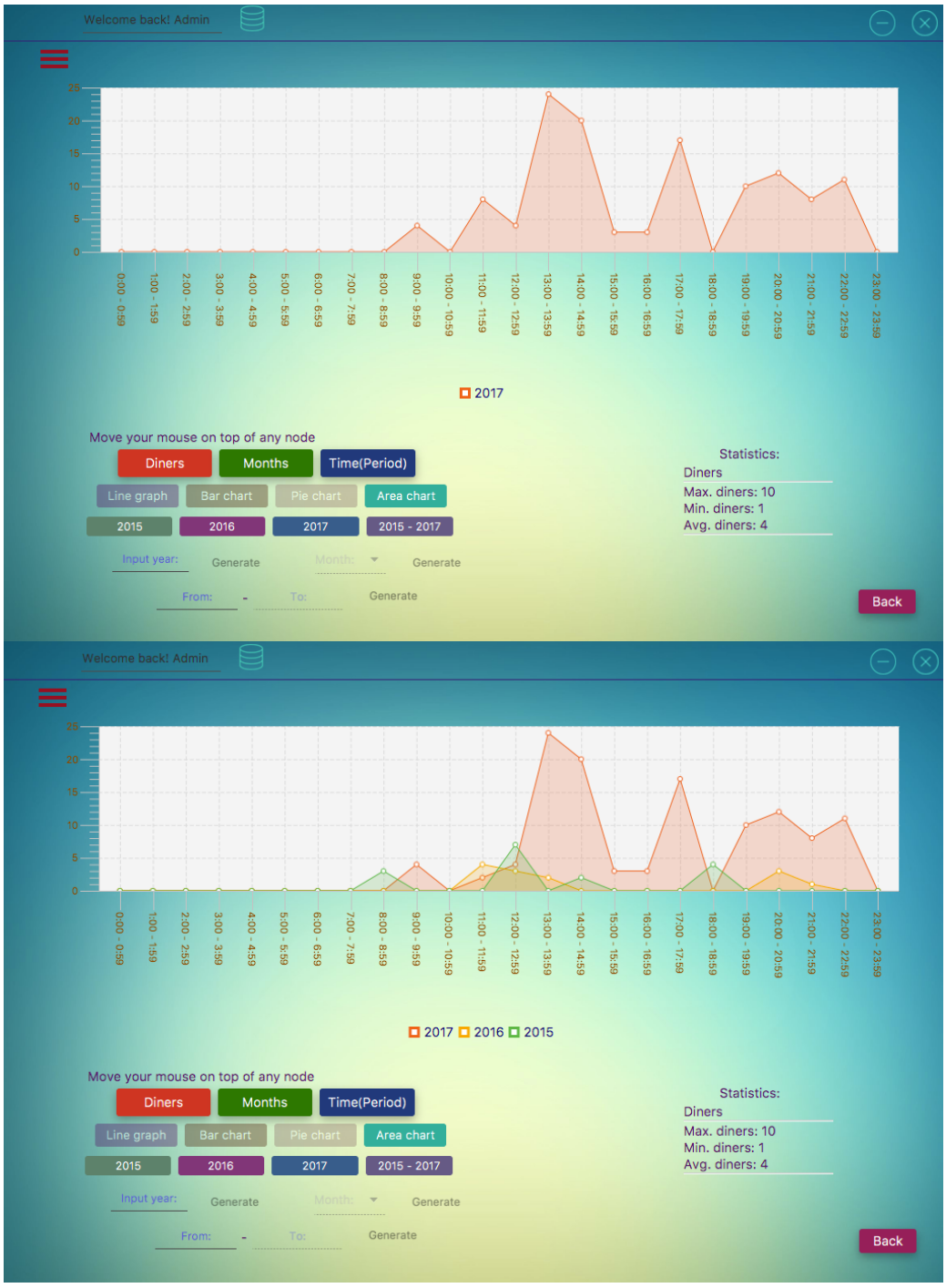



**Figure 9-9-4.** Area charts

## 4.6 User interface design

The system's window size is 1024x700. Background changed depended on the each section, all text fields, buttons, list-views and table views have a suitable size. By having all this user interface deign, I believe the system will provide better user experience.

## 4.7 Summary

In this chapter, I mentioned HLD and LLD for the system design and implementation. In addition, I had carefully discussed application walk though and added explanation at each stage. The details of application walk through has been discussed and support with system interfaces.

# Chapter 5: Testing

This chapter discusses the test cases for each scenes, and the evaluation of the system. The system has been developed by one scene at a time, which redundant the mistakes and made the implementation process smoother. I have categories my system into four sections, reservation system, estimating tables waiting time, view database records and data analysis.

## 5.1 Reservation system efficiency

This section has been considered as the most important part for my system, because a good reservation system can optimise the user booking experience with the restaurant. I had asked ten students to use my reservation system and timed it. The average time they spent to complete the task is a minute.

## 5.2 Accuracy of estimating tables waiting time

The accuracy of estimating tables waiting time was tested in the following way:

- Booked all the table and time slots, the switched scene to online self-queue. At first, the system calculated approximate waiting time depended on the maximum table time stored in the database and the first table finished time. To check the accuracy of calculation, I found the first table finished time and added the maximum table, the result is equal to the system, therefore the estimation was correct.
- Next, I joined the queue by entered contact number, name and number of diners. The queue list had instantly updated and printed out the details I entered with my queue number. On the estimating tables waiting time list, it had updated the total number of people in the queue as well.

## 5.3 View database records efficiency and stability

For the restaurant managers and staffs, view database records has played an important role. The fulfilment of the quick, clear and flitting records tasks, I had carefully tested the system. First, I repeatedly switched the table view between today's booking, all booking and staff details. From the result conducted, there is no lag appeared when I switched the view. Second, I typed few texts to the search field, the specific records had shown immediately.

## 5.4 Data analysis efficiency and stability

The data analysis efficiency and stability was tested in the following ways. I had constantly swapped the types of graph, area chart, bar chart, line graph and pie chart with different result sets, number of diners, number of records and time series. The system seems had no lag at transforming numeric data from the database to different type of graphs.

## 5.5 Summary

This chapter reflected on the testing strategies applied to my system and the results conducted. The testing strategies had explained in details, to sum up, all tested results had matched the expectations and system had no bad responds.

# Chapter 6: Conclusions

This chapter captures what I have learnt and achieved from this project, additionally, the expected and unexpected challenges that I faced provided with the solutions.

## 6.1 What I learned and achieved

I had developed an rugby statistics mobile application while I was studying computing at A-levels, at that time I only managed to implement the basic input data, store data and output data functions. In terms of interface, there were hardly any interface design.

This time, I had applied the knowledge gained from the Interaction Design and Graphical User Interfaces university modules, which how to design a good user interface, interactions with the end user, and represents the numeric data into graphs. Software Engineering module had developed my project organisation and management skills, which played the important role to complete my project on time and managed to work with other coursework.

Not only those modules had helped me, Database Systems and Procedural Programming modules had truly expanded my programming skills. I were able to do the calculation using the data from the database, search the particular data that the user requested and transferred numeric data into different formats.

Consequently, compared to the application I built at A-level, clearly there is a huge differences, not only this project had provided more functionalities but more importantly the friendly user interface design. The CSS knowledge learnt from Web Programming module brought my project's interface to the next level.

To sum up, the biggest achievement I had achieved is time management. Thought out this final year, I had one individual project, one poster, two group projects, two presentations and eleven reports/essays. Although, I had a lot of assessments but I was able to handle it and hand in all the assessments on time. Owing to that I had made a project time plan at early October before I started this project and fellow through the time plan very well.

## 6.2 Challenges that I faced

When I started to work on this project, I did not have any knowledge about JavaFX application development, even though I had basic knowledges about Java. Neither did I have much experience in user interface design and implement CSS with JavaFX application.

**Table-3 List of challenges with difficulties level**

| No. | Challenges | Difficulties | Completed |
|---|---|---|---|
| 1 | Create a basic JavaFX application |  | YES |
| 2 | Create a complex JavaFX application |  | YES |
| 3 | Well-structured database |  | YES |
| 4 | Estimate table waiting time |  | YES |
| 5 | Transfer the numeric data to Area Chart |  | YES |
| 6 | Transfer the numeric data to Bar Chart |  | YES |
| 7 | Transfer the numeric data to Line Graph |  | YES |
| 8 | Transfer the numeric data to Pie Chart |  | YES |
| 9 | Apply advance CSS to JavaFX application |  | YES |
| 10 | Filter records |  | YES |
| 11 | Update views instantly |  | YES |
| 12 | Account authentication |  | YES |

As Table-3 listed, what I found the most challenge task was No.5, transfer numeric data to area chart. This is my first time to convert the data from the database onto the graph. I had achieved this task by self-learning how to programming Java Chart/Graph.

## 6.3 Summary

This chapter has summaries the achievements and challenges while implemented my project, overall I have complete all the functional, non-functional and database requirements that discussed at Chapter 3.

# Chapter 7: Further work

This chapter describes the potential works that I can add or change to my project if I have more time to develop it.

## 7.1 What would I add and do different

Looking into my project, the system has provided:

- Rich user interfaces
- Reasonable feedbacks
- Animations
- Transitions
- Well developed functions

However, if I have more time I would add, in fact I would defiantly want to continuous developing this restaurant reservation system after I graduated from Queen Mary University of London.

- Login system for restaurant bookers: Currently, there is a login system for the restaurant admins, but adding a login system for the bookers will bring the positive effects to the user:
    - During booking process system will auto-fill in first name, surname, contact number, e-mail address.
    - Can view their booked table histories.
    - Can edit upcoming booking without entering contact number and booking reference.
- Automatically send a remind booked table message via e-mail: Currently the system will send a confirmation booking message via e-mail after the user booked a table. I believe it would be more user friendly if system can send a remind upcoming event message via e-mail to user.
- Automatically send a table ready message if diners are in the queue: Currently the system will only display the estimating table waiting time, but I believe it would be more convenient to the user if system can send a notify mobile phone text message or short e-mail message to user. User will not need to remember the waiting time and can use the waiting time to do other tasks.

## 7.2 Summary

This section has demonstrated the potential works I would love to added to my project and explained the effects will bring to the bookers and admins.

# References / bibliography

[1] World Internet Users Statistics by Regions Dec. 2016, Internet World Stats
http://www.internetworldstats.com/stats.htm

[2] DBMS - database management system, Vangie Beal, 2005
http://www.webopedia.com/TERM/D/database_management_system_DBMS.html

[3] NoSQL DBMS (Not only SQL database management system), Margaret Rouse, 2015
http://searchdatamanagement.techtarget.com/definition/NoSQL-DBMS-Not-only-SQL-database-management-system

[4] What is data analysis?, Molly Galetto, 2017
https://www.ngdata.com/what-is-data-analysis/

[5] Data Analytics (DA), Margaret Rouse, 2016
http://searchdatamanagement.techtarget.com/definition/data-analytics

[6] The JavaFX architecture, Manoj Debnath, 2016
http://www.developer.com/java/data/what-is-javafx-an-introduction.html

[7] What is CSS and Where Is It Used?, Jennifer Kyrnin, 2016
https://www.thoughtco.com/what-is-css-3466390

Using JavaFX Charts, Alla Redko, 2014
https://docs.oracle.com/javafx/2/charts/jfxpub-charts.htm

Introduction to JavaFX Charts, Alla Redko, 2014
https://docs.oracle.com/javafx/2/charts/chart-overview.htm#CJAHHJCB

Line Chart, Alla Redko, 2014
https://docs.oracle.com/javafx/2/charts/line-chart.htm#CIHGBCFI

Bar Chart, Alla Redko, 2014

https://docs.oracle.com/javafx/2/charts/bar-chart.htm#CIHJFHDE

Pie Chart, Alla Redko, 2014

https://docs.oracle.com/javafx/2/charts/pie-chart.htm#CIHFDADD

Area Chart, Alla Redko, 2014

https://docs.oracle.com/javafx/2/charts/area-chart.htm#CIHCFGBA

JFoenix

https://github.com/jfoenixadmin/JFoenix

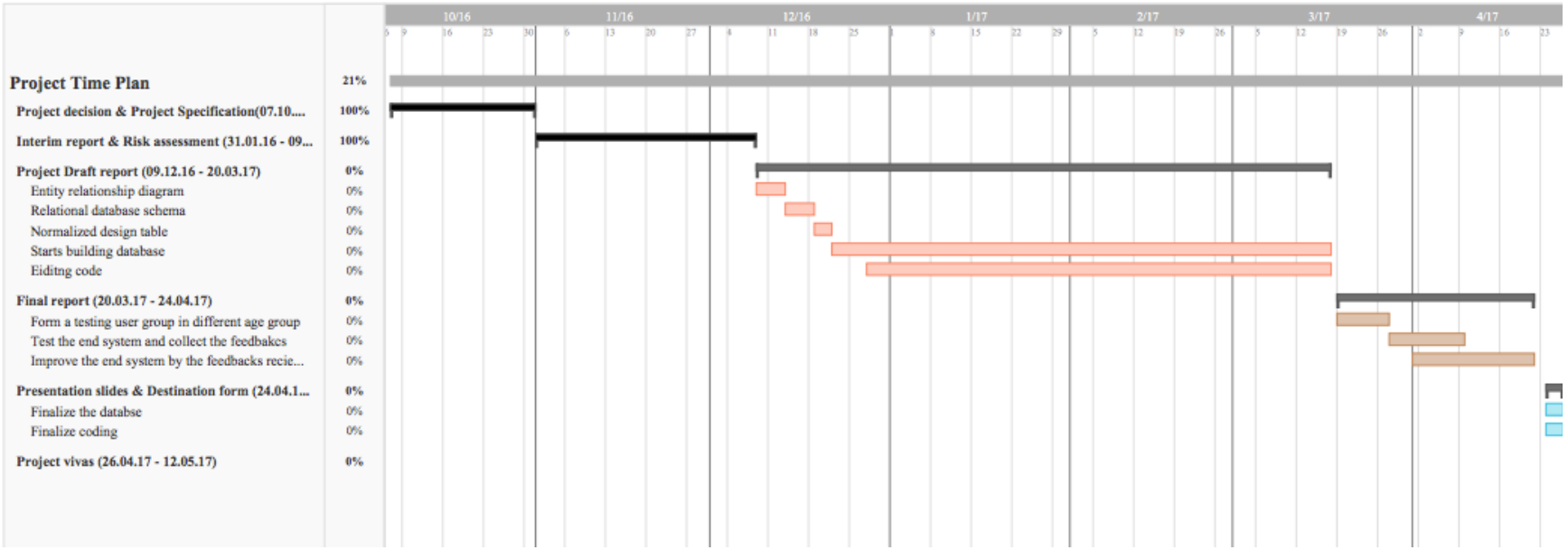
# Appendices

This section includes information about supporting materials and Gantt chart.

## Supporting materials

The full source code and executable file for this project can be found under the following directories:

- **Source code**: see folder "./src/RDBMSA"
- **Css file:** see folder "./src/CSS"
- **Executable file:** see folder "./dist/Reservation_database_system_with_analysis.jar"
- **Diagrams:** see folder "./Documents/"
    - **Walk-through diagrams:** see folder "./Documents/walk-through"
    - **State/Class/Use case diagrams:** see folder "./Documents/diagrams"
    - **JavaFX diagrams:** see folder "./Documents/FxDiagrams"


10/16
11/16
12/16
1/17
2/17
3/17
4/17
Project Time Plan 21%
Project decision & Project Specification(07.10.... 100%
Interim report & Risk assessment (31.01.16 - 09... 100%
Project Draft report (09.12.16 - 20.03.17) 0%
Entity relationship diagram 0%
Relational database schema 0%
Normalized design table 0%
Starts building database 0%
Eiditng code 0%
Final report (20.03.17 - 24.04.17) 0%
Form a testing user group in different age group 0%
Test the end system and collect the feedbakes 0%
Improve the end system by the feedbacks recie... 0%
Presentation slides & Destination form (24.04.1... 0%
Finalize the databse 0%
Finalize coding 0%
Project vivas (26.04.17 - 12.05.17) 0%